Cytoscape 3

As Building Blocks for New Network Biology Applications

Keiichiro Ono (kono@ucsd.edu)
Johannes Rushcheinski (jruscheinski@ucsd.edu)
Peng-Liang Wang (plwang@bioeng.ucsd.edu)
University of California, San Diego Department of Medicine

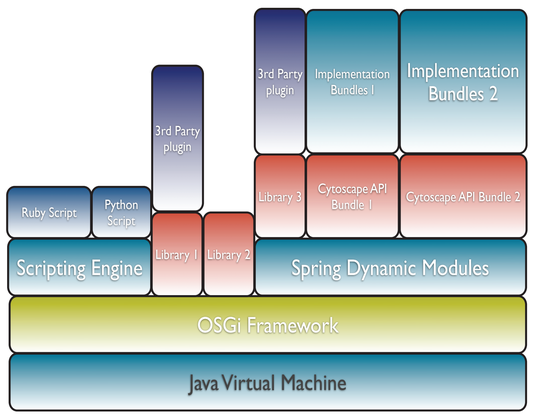

Fig 1. Basic Architecture of Cytoscape 3

Introduction

Cytoscape 3 is a new version of Cytoscape with a cleaner and far more modular architecture than the 2.x series. Although the Cytoscape 2.x series is the de-facto standard network visualization software in systems biology research we felt the urgent need to fix some fundamental problems with the 2.x architecture. The strategies chosen for 3.x allow easily interchangeable implementations, e.g. of the network model and the rendering engine as well as a tighter and better controlled API that changes in a backwards compatible manner which reduces the burden on plug-in authors. Many of the Cytoscape 2.x APIs have been rethought simplified and improved in various ways in Cytoscape 3. Separation between interfaces and implementations are now enforced in 3.x rather than merely implied as in 2.x allowing us to continually improve the implementations without affecting functionality or backwards compatibility. In summary, we now utilize modern modular Java strategies to provide us with various benefits that we would have liked to have in the 2.x series.

In this poster, we are going to introduce three case studies how to implement new features in Cytoscape 3 that would have been difficult to implement in Cytoscape 2.x. These case studies shows how to take advantage of the new architecture of Cytoscape 3.

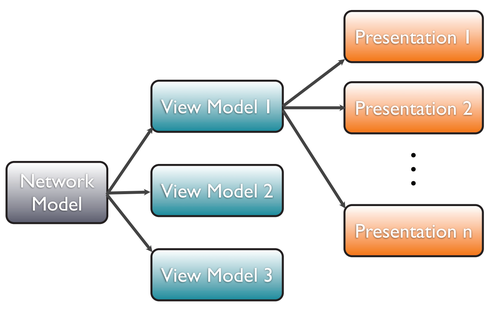

Fig. 2 Data visualization pipeline in Cytoscape 3

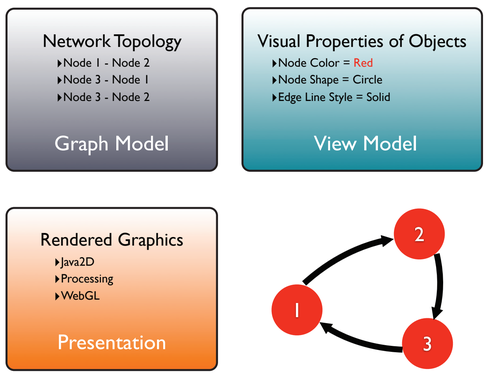

Fig. 3 Visualization modules in Cytoscape 3

Architecture of Cytoscape 3

Cytoscape 3 Architecture

Cytoscape 3 is composed of a collection of OSGi bundles (fig.1). These are loosely speaking software modules that provide and consume services. Typically there are 2 categories: interface (public) and implementation (private). OSGi helps enforce module boundaries and to keep implementation code private and inaccessible from the outside of an implementation bundle. Configuration of services and dependency injection in class creation is done with Spring XML files. This allows easy swapping out of differing implementations of an interface w/o the rewriting of any code. Apart from the greater modularity we also gain the advantage that different modules may use different versions of Java libraries w/o any potential version clashes. Using OSGi also allows us to have a much more tightly controlled and far smaller public API and better API versioning. (We use semantic versioning.)

Basic concepts of Cytoscape 3 plugin development

Cytoscape supports a so-called “plug-in architecture”. What this means is that its functionality can be extended at runtime by loading external modules. Cytoscape 3 supports 2 types of plugins. 1) Simple plugins, and 2) OSGi bundle plugins. Simple plugins can be developed with as little as a the Cytoscape plugin API jar file, a source code editor and the javac command whereas OSGi bundle plugins require more knowledge and infrastructure. Typically OSGi bundle plugins start with a Maven archetype and then continue to use Maven as the build tool. The main reasons that developers may choose to use the more complex OSGi bundle style of plugin is that they can then export their own API and that these more complex types of plugins protect plugins from Java library version conflicts. Due to the higher complexity in creating OSGi bundle plugins most plugin creators will probably choose the simple plugin style.

Hello World Plugin for Cytoscape 3

Here we’re discussing how to create a very simple Cytoscape 3 plugin, a Hello World plugin, which adds a menu item under the Plugins pull-down menu in Cytoscape. We could build such a plugin in two ways. (1) As a simple plugin. In this case, all we need are two classes, a HelloWorld class, which extends CyPlugin interface (Plugin-api bundle) and a HelloWorldAction class, which implements the CyAction interface. Then we register the action through CyPluginAdapter. (2) As an OSGi bundle plugin. This is more involved than the former. First, use a Maven archtype to create a Maven project – HelloWorld. Create a HelloWorldAction class, which implements the CyAction interface. Through the Spring configuration, we create a HelloWorldAction bean, and then register the bean (CyAction) as an OSGi service. Either way, after the HelloWorld plugin is deployed, we should see the menu-item “Hello World” under Plugins menu in Cytoscape 3.

Case Study 1: Workflow as a Collection of Tasks

Case Study 2: Custom Rendering Engine

Fig.4 Treemap representation of hierachical data. (National Visualization and Analytics Center. 2005)

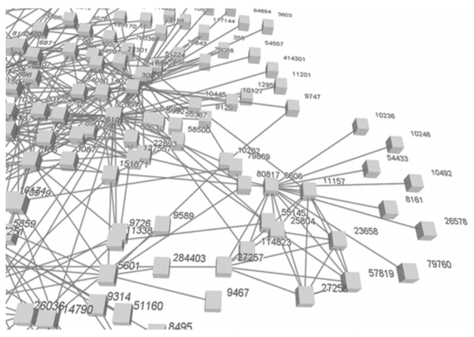

Fig. 5 Experimental visualizer based on Processing visualization programming environment

Background

Cytoscape 3 has clear separations between network model, view model, and actual visualization (presentation). This means Cytoscape can have multiple visualizations for a data set (fig. 1, 2). The current visualization engine (rendering engine) is sufficient for many network visualization needs, but for some use cases, such as more complex pathway diagram visualization using the SBGN [3] standard or browsing a huge network with a highly interactive user interface using GPU support, it is not enough to satisfy those needs. As a complement to the current rendering engine, developers can add additional visualizers to render their data sets in other ways. This multiple rendering engine mechanism is built-in for Cytoscape 3, and it is relatively easier to implement than in Cytoscape 2. In addition, the concept of view model is not only for network data. Developers can implement view models for table data (attributes). This allows developers to visualize table data as interactive charts.

Applications

A typical example of a new network visualizer is Treemap [4]. Treemap is a visualization technique to display hierarchical data by using nested rectangles(fig. 3). Many network data sets contain a hierarchical structure and Treemap is a good addition to the current network rendering engine to understand the overall structure of the data.

One of the missing features in the current rendering engine is drawing complex human-curated pathway diagrams. Supporting SBGN is a solution for this problem, but it is very hard to implement everything from scratch. CellDesigner [13] is widely used application in the SBGN community and it has a complete editor for SBGN. Once it has been released as open source software, Cytoscape can use its editor module as a new rendering engine for pathways.

Implementation

In general, developing a new visualizer in Cytoscape 3 is equivalent to implementing interfaces in the presentation API module. The main component in that module is called RenderingEngine. It is a stateless object drawing graphics based on an associated view model. Rendering engine developers should write code to interpret visual properties (a collection of visual states, such as node color is red, edge width is 10, etc.) into actual graphics. Figure 4 is an experimental new rendering engine based on the Processing visualization programming environment [5]. This is a new visualizer based on the OpenGL canvas in Processing for large network visualization. Source code is available from the GitHub repository [6].

Case Study 3: Graph Database Backend

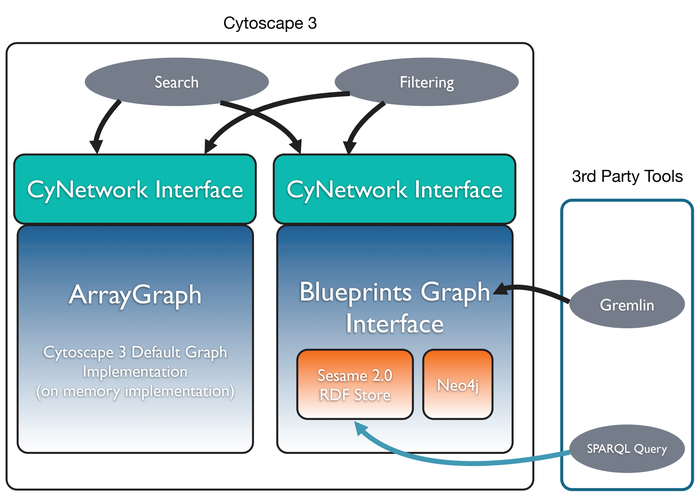

Fig. 6 Multiple graph implementations in Cytoscape 3. By wrapping 3rd party graph implementations with ***CyNetwork*** interface, users can access network data from both Cytoscape and 3rd party tools.

Background

Currently, Cytoscape has only one implementation of the graph model called ArrayGraph. This is an in-memory implementation of the CyNetwork interface. To analyze and visualize huge networks, typical work flow usually starts with querying and filtering the network and then creates sub networks. It is not efficient to load the entire network into memory in such work flows. In addition, querying huge networks is a popular research field and many tools are available from the semantic web and social science communities. To utilize those tools in Cytoscape 3, wrapping popular graph implementations, including RDF triple stores and disk-based graph databases, is a good approach.

Applications

Although Cytoscape is focusing on biological data sets, its core is problem domain independent. In social science, there are many data sets containing millions of nodes and edges. To handle such networks, it is better to use a disk-based data model instead of in-memory graph implementations. Also, building a new infrastructure in Cytoscape to handle huge network data set is a good preparation for handling genetic interaction data sets, which potentially contains millions of edges.

These days, biological public data sets are available in the RDF format [x]. If Cytoscape provides native support for existing triple stores like Sesame [], users can use many of existing tools from the semantic web community.

Implementation

Today we have many choices of disk-based graph database. Neo4j [x] is one of the most popular graph database implementations. To integrate this graph implementation in Cytoscape 3, using Tinkerpop Blueprints layer [x] is a good option. Blueprints is a framework to provide a common interface for graph databases, such as Neo4j, OrientDB[], and Sesame 2.0. This is a good strategy to avoid vendor lock-in. The integration looks like figure 6. By wrapping the Blueprints implementations with the CyNetwork interface, users have access to the network data from the standard Cytoscape interface and third party frameworks like Gremlin [x].

Advanced Topics

Using Cytoscape 3 Components in Other Applications

Since the Cytoscape 3 application is just a collection of bundles, it is relatively easy to reuse its components in other applications. Although Cytoscape 3 will be released as a desktop application first, you can create other applications based on Cytoscape 3 bundles.

Command Line Application

Due to its modular nature, it is possible, for example, to create command-line utilities for the batch processing of large networks and/or the generation of network image data files. Many other applications are conceivable only limited by the requirements and creativity of a possible utility author.

Server Back-end / Cloud Deployment

If developers use Cytoscape 3 components in their server-side applications, they can take advantage of existing Cytoscape 3 features. Standard network file readers (GML, XGMML, SIF, GraphML, KGML, etc.) or layout algorithms are rather common components for most of the graph analysis/visualization applications and by using the same data/view model used in Cytoscape 3, they do not have to implement their own version of these functions and they can focus on their own functionality.

Separation of the presentation layer is also an advantage in web application development. In the model layer, developers can use Cytoscape components as a part of their server-side Java application, and presentation can be implemented with the latest browser technology, including HTML 5 or WebGL [14] (fig. 7).

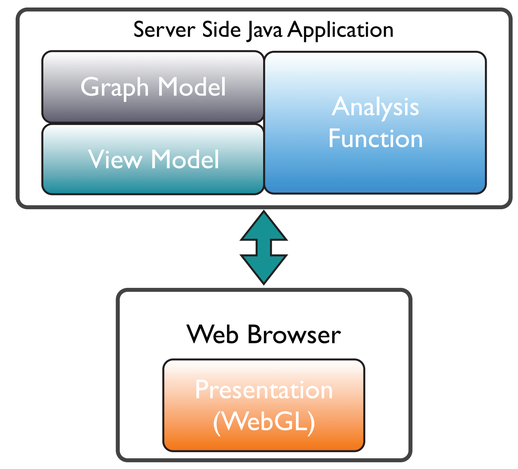

Fig. 7 Cytoscape Components can be used in serverside.

References

1. Apache Felix. http://felix.apache.org/site/index.html
2. Equinox OSGi implementation. http://eclipse.org/equinox/
3. Systems Biology Graphical Notation (SBGN) project. http://www.sbgn.org/
4. Treemaps for space-constrained visualization of hierarchies. Shneiderman. B. http://www.cs.umd.edu/hcil/treemap-history/index.shtml
5. Processing programming environment. http://processing.org/
6. Processing-Based Visualizer for Cytoscape 3. https://github.com/kozo2/processing-renderer/wiki
7. OpenRDF. http://www.openrdf.org/
8. Neo4j: The Graph Database. http://neo4j.org/
9. TinkerPop Blueprints property graph model interface. https://github.com/tinkerpop/blueprints
10. Gremlin graph traversal language https://github.com/tinkerpop/gremlin/wiki
11. Resource Description Framework (RDF). http://www.w3.org/RDF/
12. OrientDB graph database. http://www.orientechnologies.com/
13. CellDesigner: A modeling tool of biochemical networks. http://www.celldesigner.org/
14. WebGL: OpenGL ES 2.0 for the Web. http://www.khronos.org/webgl/

Web: www.cytoscape.org Twitter: cytoscape